Advanced Crash Course in Supercomputing: Supercomputers, Batch Scripts, and Smoky

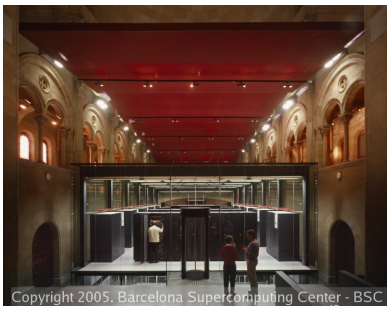
OLCF
Oak Ridge Leadership Computing Facility

Rebecca Hartman-Baker
Oak Ridge National Laboratory
hartmanbakrj@ornl.gov

U.S. DEPARTMENT OF ENERGY
Oak Ridge National Laboratory
Managed by UT-Battelle for the Department of Energy

Outline

I. Supercomputers
II. Batch Scripts
III. Using Smoky



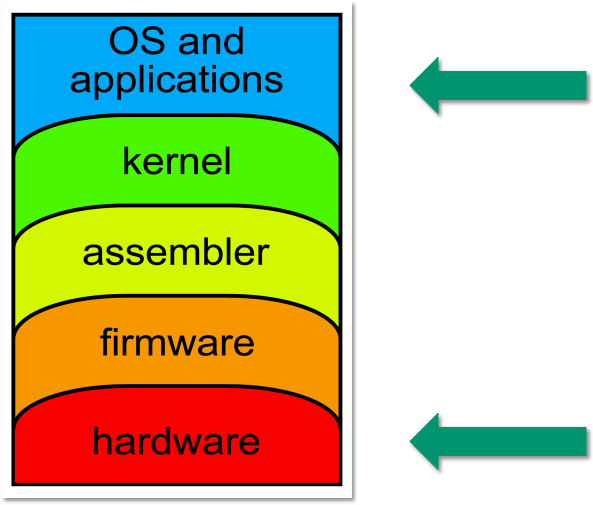
I. SUPERCOMPUTERS

Mare Nostrum, installed in Chapel Torre Girona, Barcelona Supercomputing Center. By courtesy of Barcelona Supercomputing Center -- http://www.bsc.es/

I. Supercomputers

- Computer Architecture 101
- OLCF Facts and Figures
- XT5 Architecture

Computer Architecture 101



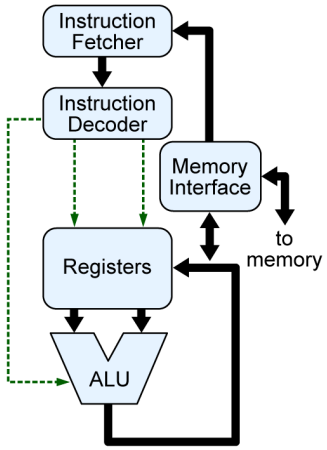
Source: http://en.wikipedia.org/wiki/Image:Computer_abstraction_layers.svg (author unknown)

Computer Architecture 101

- Processors
- Memory
  - Memory Hierarchy
  - TLB
- Interconnects
- Glossary

Computer Architecture 101: Processors

- CPU performs 4 basic operations:
  - Fetch
  - Decode
  - Execute
  - Writeback



Source: http://en.wikipedia.org/wiki/Image:CPU_block_diagram.svg

CPU Operations

- Fetch
  - Retrieve instruction from program memory
  - Location in memory tracked by program counter (PC)
  - Instruction retrieval sped up by caching and pipelining
- Decode
  - Interpret instruction by breaking into meaningful parts, e.g., opcode, operands
- Execute
  - Connect to portions of CPU to perform operation, e.g., connect to arithmetic logic unit (ALU) to perform addition
- Writeback
  - Write result of execution to memory

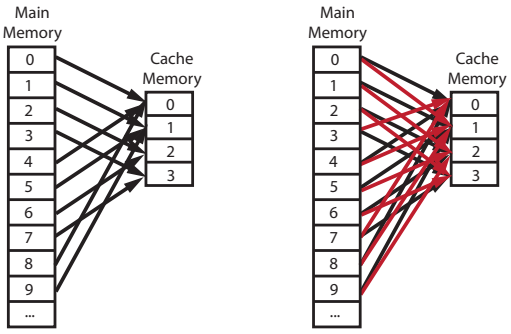
Computer Architecture 101: Memory

- Hierarchy of memory
  - Fast-access memory: small (expensive)
  - Slower-access memory: large (less expensive)
- Cache: fast-access memory where frequently used data stored
  - Reduces average access time
  - Works because typically, applications have locality of reference
  - Cache in XT4/5 also hierarchical
- TLB: Translation lookaside buffer
  - Used by memory management hardware to improve speed of virtual address translation

Cache Associativity

- Where to look in cache memory for copy of main memory location?
  - Direct-Mapped/ 1-way Associative: only one location in cache for each main memory location
  - Fully Associative: can be stored anywhere in cache
  - 2-way Associative: two possible locations in cache
  - *N*-way Associative: *N* possible locations in cache
- Doubling associativity (1 → 2, 2 → 4) has same effect on hit rate as doubling cache size
- Increasing beyond 4 does not substantially improve hit rate; higher associativity done for other reasons

Cache Associativity: Illustration

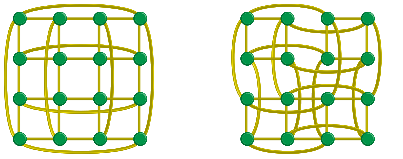


Direct-Mapped Cache

2-Way Associative Cache

Computer Architecture 101: Interconnects

- Connect nodes of machine to one another
- Methods of interconnecting
  - Fiber + switches and routers
  - Directly connecting
- Topology
  - Torus
  - Hypercube
  - Butterfly
  - Tree



2-D Torus

Hypercube

Computer Architecture 101: Glossary

- *SSE (Streaming SIMD Extensions):* instruction set extension to x86 architecture, allowing CPU to work on multiple instructions in single clock cycle
- *DDR2 (Double Data Rate 2):* synchronous dynamic random access memory, operates twice as fast as DDR1
  - DDR2-*xyz*: performs *xyz* million data transfers/second
- *Dcache:* cache devoted to data storage
- *Icache:* cache devoted to instruction storage
- *STREAM:* data flow

OLCF Facts and Figures

	Jaguarpf	Kraken	Gaea
Compute Nodes	18,772	9408	2576
Processor	2.3 GHz AMD Opteron Dual Hex-Core	2.3 GHz AMD Opteron Dual Hex-Core	2.1 GHz AMD "Magny-Cours" 12-core
Memory	16 GB/node DDR2-800	16 GB/node DDR2-800	64 GB/node DDR3
Network	Cray SeaStar 2, 3-D Torus	Cray SeaStar 2, 3-D Torus	Cray Gemini, 3-D Torus
Peak	2.3 PF	1.17 PF	260 TF

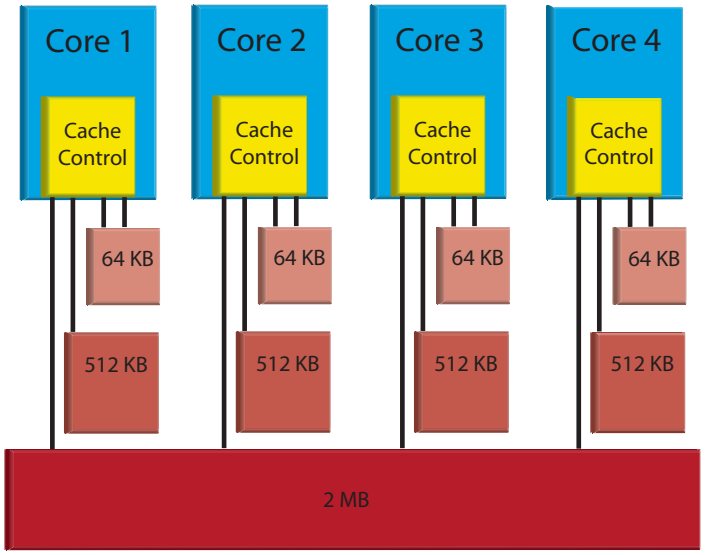
XT5 Architecture

- Hardware
  - Processors
  - Memory
    - Memory Hierarchy
    - TLB
  - System architecture
  - Interconnects
- Software
  - Operating System Integration
  - CNL vs Linux

XT5 Hardware: Processors

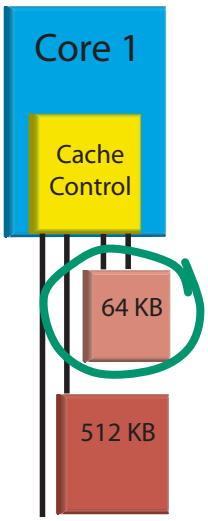
- On each node:
  - Two 6-core AMD Istanbul Opteron processors, 2.2 GHz
  - 16 GB shared memory
  - 125 GF peak performance

Quad Core Cache Hierarchy



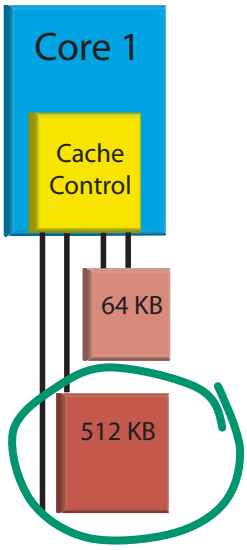
L1 Cache

- Dedicated
- 2-way associativity
- 8 banks
- 2 x 128-bit loads/cycle



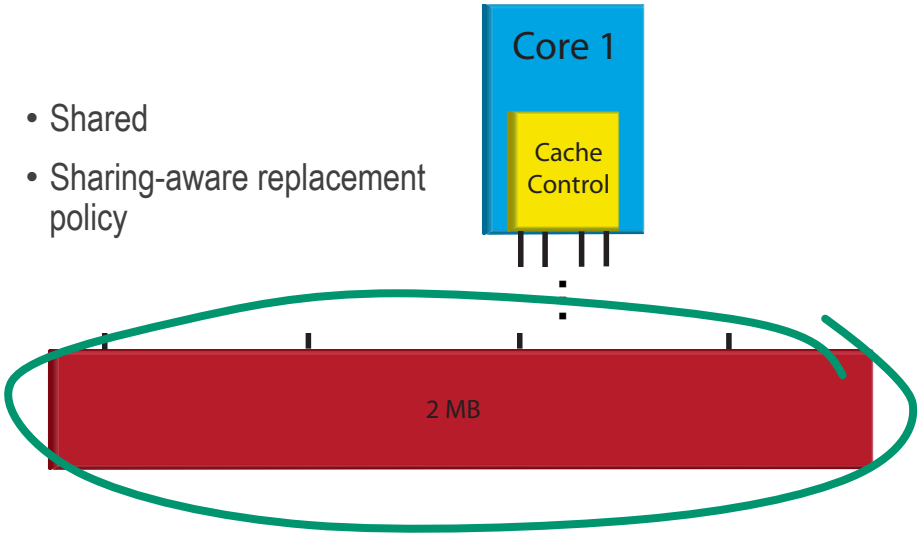
L2 Cache

- Dedicated
- 16-way associativity



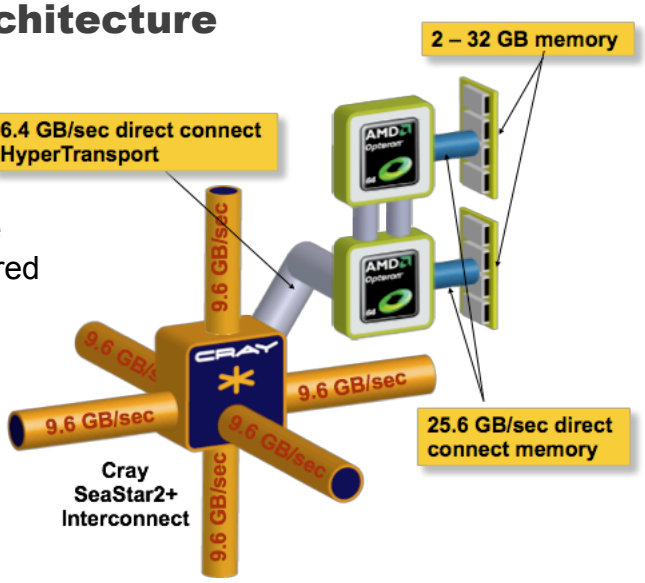
L3 Cache

- Shared
- Sharing-aware replacement policy



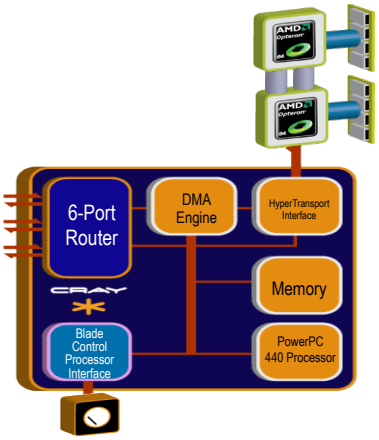
Cray XT5 Architecture

- 8-way SMP
- > 70 Gflops/node
- Up to 32 GB shared memory/node
- OpenMP support



Cray SeaStar2 Architecture

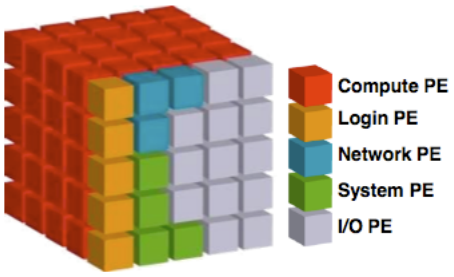
- Router connects to 6 neighbors in 3-D torus
  - Peak bidirectional BW 7.6 GB/s; sustained 6 GB/s
  - Reliable link protocol with error correction and retransmission
- Communications Engine: DMA Engine + PPC 440
  - Together, perform messaging tasks so AMD processor can focus on computing
- DMA Engine and OS together minimize latency with path directly from app to communication hardware (without traps and interrupts)



System Architecture

- 18,688 nodes in 200 cabinets
- 3-D torus interconnect
- 4352 square feet floorspace
- 1750 W/sq. ft. power consumption
- Liquid cooling system

Software Architecture



Service Partition
Specialized Linux nodes

- CLE microkernel on compute nodes
- Full-featured Linux on service nodes
- Software architecture eliminates jitter and enables reproducible runtimes
- Even large machines can reboot in < 30 mins, including filesystem

Software Architecture

- *Compute PE (processing element):* used for computation only; users cannot directly access compute nodes
- *Service PEs:* run full Linux
  - *Login:* users access these nodes to develop code and submit jobs, function like normal Linux box
  - *Network:* provide high-speed connectivity with other systems
  - *System:* run global system services such as system database
  - *I/O:* provide connectivity to GPFS (global parallel file system)

CLE vs Linux

- CLE (Cray Linux Environment) contains subset of Linux features
- Minimizes system overhead because little between application and bare hardware

Life at an HPC Center

- Work
- People
- Careers

Resources: Computer Architecture 101

- Wikipedia articles on computer architecture: http://en.wikipedia.org/wiki/Computer_architecture , http://en.wikipedia.org/wiki/CPU , http://en.wikipedia.org/wiki/CPU_cache , http://en.wikipedia.org/wiki/DDR2_SDRAM , http://en.wikipedia.org/wiki/Microarchitecture , http://en.wikipedia.org/wiki/SSE2 , http://en.wikipedia.org/wiki/Streaming_SIMD_Extensions
- Heath, Michael T. (2007) *Notes for CS 554, Parallel Numerical Algorithms*, http://www.cse.uiuc.edu/courses/cs554/notes/index.html

Resources: Cray XT5 Architecture

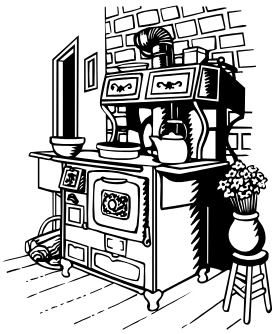
- Local machines
  - Jaguarpf: http://www.olcf.ornl.gov/computing-resources/jaguar/
  - Kraken: http://www.nics.tennessee.edu/computing-resources/kraken
  - Gaea: http://www.ncrc.gov/computing-resources/gaea/
- XT4 Architecture
  - Hartman-Baker, Rebecca (2008) *XT4 Architecture and Software*, available at http://www.nccs.gov/wp-content/training/2008_users_meeting/4-17-08/using-xt44-17-08.pdf

II. BATCH SCRIPTS

Soft Batch Cookies. From http://www.kelloggconvenience.com/Resources/Soft_Batch-Home-PBpouch.jpg

II. Batch Scripts

- Batch system and Scheduling
- Concepts
- Useful commands
- Further help



Batch System and Scheduling

- Supercomputer: powerful computer consisting of many interlinked CPUs
- Users competing for computational resources
- How to launch and schedule jobs fairly?
- Job can run without user presence
- Must not allow one user to hog resources

Batch System

- Batch system accepts input jobs into queue and launches them when resources available
- Many machines use batch system PBS (*P*ortable *B*atch *S*ystem)
- PBS developed for NASA in 1990s

Scheduler

- Scheduler decides when jobs can be run based on scheduling policies, e.g. user priority, length of job, number of nodes requested, length of time in queue
- Many machines use Maui Scheduler
- Maui Scheduler extensively developed, supported by large segment of computation community including U.S. Dept. of Energy, NCSA

(source: www.the-hawaii-vacation-guide.com)

Concepts

- Limits for walltime and number of processors, so if request exceeds limits, job automatically rejected
- Scheduler rules complicated, but generally, “smaller” jobs run first
- Size of job is function of number of processors and estimated time
- You provide info about number of processors you want and estimate of time job will run

Concepts

- Strategies:
  - Like inverse of “The Price Is Right,” give lowest estimate possible, without going under true time needed (always good strategy)
  - Use fewer processors if possible (not always good strategy)
- If you reach end of estimated time, PBS will terminate your job!
- Write script that tells PBS what to do when job is launched

Concepts

- Shell Script format:
  - First, a line invoking the scripting language:
    ```
    #!/bin/csh
    ```
  - Next, embedded PBS commands, e.g.
    ```
    #PBS -l walltime=00:10:00,nodes=2:ppn=2
    #PBS -q workq
    ```
    (the shell script interprets these as comments, but PBS understands they are PBS commands)
  - Then, environment variable initialization, e.g.
    `setenv MYMAINDIR /home/hqi/hello` (sets variable `MYMAINDIR` to `/home/hqi/hello`)
    `setenv PROG $MYMAINDIR/prog` (sets `PROG` to `/home/hqi/hello/prog`)

Concepts

- Shell script format (continued):
  - Then, shell script and regular Linux commands, e.g.
    ```
    if (-e $OUTF) mv $OUTF $OUTF.old
    ```
    (meaning that if file called `$OUTF` exists, rename it to `$OUTF.old`)
  - Finally, run job:
    ```
    mpirun -np $NP $PROG < $INFILE > $OUTF
    ```
- To launch job:
  - Make script executable*: `chmod u+x myscript`
  - `qsub myscript`

***Not necessary on some systems**

Useful Commands (PBS)

- `#PBS -l walltime=hh:mm:ss,nodes=n:ppn=p` This tells PBS how much walltime you request (where `hh:mm:ss` replaced by appropriate number of hours, minutes, and seconds), how many *dual processor* nodes you want (replace `n` with appropriate number), *and how many processors per node (1, 2, 3, or 4)*
- `#PBS -q workq` Which queue to use (in this case, queue called workq)
- `#PBS -V` Export all environment variables to batch job (good practice to do this)
- `#PBS -m be` Sends you e-mail at beginning and end of job

Useful Commands (Shell Scripting)

- `set echo` Print out commands as they are executed (useful for debugging script)
- `setenv A B` or `export A=B` Sets environment variable `A` to `B`
- `$A` value of `A`
- `mpirun -np $NP $PROG < $INPUT`
  `mpirun` (sometimes `mpiexec`, or on proprietary systems, `aprun`, `poe`, etc.) is executable that launches parallel jobs on multiple processors; `-np` is flag indicating number of processors used in run

  *NOTE: some implementations do not require input redirection (<)

Further Help

- NCSA Cobalt Documentation: Running Jobs http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/SGIAltix/Doc/Jobs.html
- The C Shell tutorial http://www.eng.hawaii.edu/Tutor/csh.html
- DuBois, Paul. *Using csh & tcsh,* O'Reilly & Associates, 1995.
- Newham, Cameron and Bill Rosenblatt. *Learning the bash Shell,* O'Reilly & Associates, 1998.

Bibliography/Resources

- About OpenPBS http://www.openpbs.org/about.html
- Maui Scheduler http://www.supercluster.org/maui/

III. USING SMOKY

Sunset from Clingmans Dome, Great Smoky Mountains National Park, photo available at http://www.nps.gov/grsm/photosmultimedia/index.htm

III. Using Smoky

- About Smoky
- Logging In
- Compiling
- Software Environment
- Running Jobs

About Smoky

- Development cluster, comparable to larger NCCS machines
- Used for application development
- 80 node Linux cluster
- Each node consists of four quad-core 2.0 GHz AMD Opteron processors, with 32 GB memory (2GB/core)
- Gigabit ethernet network with infiniband interconnect

Logging in to Smoky

- Use ssh to connect

  `ssh` *`username`*`@smoky.ccs.ornl.gov`

- Authentication using one-time passwords from RSA SecurID key fob
- X11 Tunneling: use `-X` (or on a Mac, `-Y`) option with ssh

Compiling on Smoky

- Three compiler suites available on smoky:
  - PGI (default)
  - Pathscale
  - GCC
- MPI compilers (wrappers to compiler independent of programming environment)
  - `mpicc` (C compiler)
  - `mpiCC` (C++ compiler)
  - `mpif77` (Fortran 77 compiler)
  - `mpif90` (Fortran 90 compiler)

Software Environment on Smoky

- Suppose I need to use GNU C++ compiler to compile my code
- Suppose I also want to link with the PETSc library
- On most systems, would need to change paths in makefiles each time I port to new system
- Would need to make sure to point to GNU compiler and proper build of PETSc
- What happens if I discover that I need a different compiler? Go back and change everything again

Software Environment on Smoky

- Modules allow dynamic modification of user environment with modulefiles
- Can switch from PGI to GNU and back again with simple command
- Can load proper version of PETSc automatically, based on compiler loaded

Software Environment on Smoky: Modules

- Software is loaded or swapped using modules
- Allows software, libraries, paths, etc. to be cleanly entered into and removed from your programming environment
- Conflicts are detected and loads that would cause conflicts are not allowed

Software Environment on Smoky: Modules

Command	Definition	Example
`module load my_module`	Loads module *my_module*	`module load petsc`
`module swap first_module second_module`	Replaces *first_module* with *second_module*	`module swap PE-pgi PE-gnu`
`module help`	Lists available commands and usage	
`module list`	Lists all modules currently loaded	
`module avail [name]`	Lists all modules [beginning with *name*]	`module avail gcc`

Running Jobs on Smoky

- Login node: node you log in to
  - Edit files
  - Code compilation
  - Data backup
  - Job submission
- Compute nodes
  - Where jobs run
  - Access managed by PBS
  - Scheduling by Moab

Nice Job Script for Smoky

```
#PBS -V
#PBS -j oe
#PBS -A STF006
#PBS -N loadbal
#PBS -l walltime=00:10:00,nodes=1:ppn=16
export CURRDIR="/ccs/home/hqi/hello"
export SCRDIR="/tmp/work/hqi"
export EXEC="hello"
export INPUT_FILE="hello_input"
cp $CURRDIR/$EXEC $SCRDIR
cp $CURRDIR/$INPUT_FILE $SCRDIR
cd $SCRDIR
date
mpirun -n 16 ./$EXEC < $INPUT_FILE
date
```


Resources/Bibliography

- Smoky webpage
  http://www.nccs.gov/computing-resources/smoky/
- NCCS Modules webpage
  http://www.nccs.gov/user-support/general-support/modules/